siunitx — A comprehensive (SI) units package*

Joseph Wright†

Released 2009/03/29

Abstract

Typesetting values with units requires care to ensure that the combined mathematical meaning of the value plus unit combination is clear. In particular, the SI units system lays down a consistent set of units with rules on how these are to be used. However, different countries and publishers have differing conventions on the exact appearance of numbers (and units).

The siunitx package provides a set of tools for authors to typeset numbers and units in a consistent way. The package has an extended set of configuration options which make it possible to follow varying typographic conventions with the same input syntax. The package includes automated processing of numbers and units, and the ability to control tabular alignment of numbers.

A number of LaTeX packages have been developed in the past for formatting units: SIunits, SIstyle, unitsdef, units, fancyunits and fancynum. Support for users of all of these packages is available as emulation modules in siunitx. In addition, siunitx can carry out many of the functions of the dcolumn, rccol and numprint packages.

Contents

1 Introduction 2

2 Installation 2

3 siunitx for the impatient 3

4 Using the siunitx package 3

4.1 Loading the package 3

4.2 Numbers 3

4.3 Units: as arguments 4

4.4 Units: free-standing 5

5 The unit macros 5

5.1 Creating new macros 8

6 The key–value control system 9

6.1 Detecting fonts 10

6.2 Output font families 12

6.3 Parsing numbers 13

6.4 Post-processing numbers . . 14

6.5 Printing numbers 16

*This file describes version v2.0alpha, last revised 2009/03/29.

†E-mail: joseph.wright@morningstar2.co.uk

6.6 Creating units 19
6.7 Using units 19
6.8 Symbols 21

7 Usage tips and known issues 22
7.1 Ensuring text or maths output 22
7.2 Using a comma as a separator 23

Change History 23

Index 23

1 Introduction

The correct application of units of measurement is very important in technical applications. For this reason, carefully-crafted definitions of a coherent units system have been laid down by the *Conférence Générale des Poids et Mesures*[1] (CGPM): this has resulted in the *Système International d'Unités*[2] (SI). At the same time, typographic conventions for correctly displaying both numbers and units exist to ensure that no loss of meaning occurs in printed matter.

siunitx aims to provide a unified method for LaTeX users to typeset units and values correctly and easily. The design philosophy of siunitx is to follow the agreed rules by default, but to allow variation through option settings. In this way, users can use siunitx to follow the requirements of publishers, co-authors, universities, *etc*. without needing to alter the input at all.

siunitx is intended as a complete replacement for SIunits, SIstyle, unitsdef, units, fancyunits and fancynum. As such, emulation modes are provided for all of these packages. Where possible, conventions from the existing solutions have been used here. For example, the macros `\num`, `\ang` and `\SI` act in a very similar fashion to those in existing packages.

2 Installation

The entire bundle is supplied with the TDS-ready `zip` file, `siunitx.tds.zip`. Simply unzip this into your local texmf tree and run your hash program (`texhash` for TeXLive or `initextmf -u` for MiKTeX).

To extract the package `siunitx.sty` and the configuration files from `siunitx.dtx`, two methods are available. To extract the files using the `ins` file, simply run (pdf)TeX on `siunitx.ins`. This will produce all of the package files, and also `README.txt`. To extract the files and build the documentation, run (pdf)LaTeX on `siunitx.dtx`. Three (pdf)LaTeX runs with `\write18` enabled will also build the index and table of contents in the PDF.

[1] General Conference on Weights and Measures.
[2] International System of Units.

Compilation of the package documentation requires the l3doc class, from the expl3 bundle produced by the LaTeX3 team. To compile the package documentation, you will need to get a recent version of expl3 from the LaTeX project website.

3 siunitx for the impatient

The package provides the user macros:

- `\SI[`⟨*options*⟩`]{`⟨*value*⟩`}[`⟨*pre-unit*⟩`]{`⟨*unit*⟩`}`
- `\si[`⟨*options*⟩`]{`⟨*unit*⟩`}`
- `\num[`⟨*options*⟩`]{`⟨*number*⟩`}`
- `\ang[`⟨*options*⟩`]{`⟨*angle*⟩`}`
- `\sisetup{`⟨*options*⟩`}`

plus the `S` and `s` column types for decimal alignments and units in tables. These macros are designed for typesetting units and values with control of appearance and with intelligent processing.

12 345.678 90
$1 \pm 2\mathrm{i}$
0.3×10^{45}

```
\num{12345,67890} \\
\num{1+-2i}       \\
\num{.3e45}
```

By default, all text is typeset in the current upright, serif maths font. This can be changed by setting the appropriate options: `\sisetup{font/detect/all}` will use the current font for typesetting.

4 Using the siunitx package

4.1 Loading the package

The package should be loaded in the usual LaTeX 2ε way.

```
\usepackage{siunitx}
```

The package does not use load-time options, although it does support those from version 1 of the package and predecessor packages.

4.2 Numbers

\num Numbers are automatically formatted by the \num macro. This takes one optional and one mandatory argument: \num[⟨*options*⟩]{⟨*number*⟩}. The contents of ⟨*number*⟩ are automatically formatted. The formatter removes "hard" spaces (\, and ~), automatically identifies exponents (by default marked using e or d) and adds the appropriate spacing of large numbers. A leading zero is added before a decimal marker, if needed: both "." and "," are recognised as decimal marker.

123
1234
12 345
0.123
0.1234
0.123 45
3.45×10^{-4}
-10^{10}

```
\num{123}      \\
\num{1234}     \\
\num{12345}    \\
\num{0.123}    \\
\num{0,1234}   \\
\num{.12345}   \\
\num{3.45d-4}  \\
\num{-e10}
```

4.3 Units: as arguments

\si The symbol for a unit can be typeset using the \si macro: this provides full control over output format for the unit. Like the \num macro, \si takes one optional and one mandatory argument: \si[⟨*options*⟩]{⟨*unit*⟩}. The unit formatting system can accept two types of input. When ⟨*unit*⟩ contains one or more literal items, the output is processed in the same manner as with the siunitx package. Sub- and superscripts can be input without concern over maths mode, and the tokens . and ~ are converted into an inter-unit separator and inter-unit space, respectively.[3]

$\mathrm{kg\,m/s^2}$
$\mathrm{g_{polymer}\,mol_{cat}\,s^{-1}}$

```
\si{kg.m/s^2} \\
\si{g_{polymer}~mol_{cat}.s^{-1}}
```

The second operation mode for the \si macro is an "interpretted" system, Here, each unit, SI multiple prefix and power is given a macro name. These are entered in a method very similar to the reading of the unit name in English.

```
\si{\kilo\gram\metre\per\square\second} \\
\si{\gram\per\cubic\centi\metre}        \\
\si{\square\volt\cubic\lumen\per\farad} \\
\si{\metre\squared\per\gray\cubic\lux}  \\
\si{\henry\second}
```

$\mathrm{kg\,m\,s^{-2}}$
$\mathrm{g\,cm^{-3}}$
$\mathrm{V^2\,lm^3\,F^{-1}}$
$\mathrm{m^2\,Gy^{-1}\,lx^3}$
$\mathrm{H\,s}$

[3] The standard package settings use the same value for both of these: \,.

On its own, this is less convenient than the direct method, although it does use meaning rather than appearance for input. However, the the package allows you to define new unit macros; a large number of pre-defined abbreviations are also supplied. More importantly, by defining macros for units, instead of literal values, new functionality is made available. Units may be re-defined to give different output, and handling of reciprocal values can be altered.

`\SI` Very often, numbers and values are given together. Mathematically, these form a single entity, and should be separated by a non-breaking space. The `\SI` macro combines the functionality of `\num` and `\si`, and makes this both possible and easy. The `\SI` macro takes two mandatory arguments, in addition to the optional set up argument, and a second optional argument: `\SI[`⟨*options*⟩`]{`⟨*number*⟩`}[`⟨*preunit*⟩`]{`⟨*unit*⟩`}`. The ⟨*number*⟩ and ⟨*unit*⟩ arguments work exactly like those for the `\num` and `\si` macros, respectively. ⟨*preunit*⟩ is a unit to be typeset *before* the numerical value (most likely to be a currency).

```
\SI[font/mode=text]{1.23}{J.mol^{-1}.K^{-1}} \\
\SI{.23e7}{\candela}                          \\
%\SI[per=slash]{1.99}[\$]{\per\kilogram}      \\
\SI{70}{\metre\per\second}                    \\
%\SI[per=frac,fraction=nice]{1,345}{\ampere\per\mole}
```

$1.23\,\mathrm{J\,mol^{-1}\,K^{-1}}$
$0.23 \times 10^{7}\,\mathrm{cd}$
$70\,\mathrm{m\,s^{-1}}$

4.4 Units: free-standing

5 The unit macros

The package always defines the basic set of SI units with macro names. This includes the base SI units, the derived units with special names and the prefixes. A small number of powers are also given pre-defined names. Full details of units in the SI are available on-line [1].

`\meter` The seven base SI units are always defined (Table 1). In addition, the macro `\meter` is available as an alias for `\metre`, for users of US spellings. The full details of the base units are given in the SI Brochure [2]. The SI also lists a number of units which have special names and symbols [3]: these are listed in Table 2.

In addition to the official SI units, siunitx also provides macros for a number of units which are accepted for use in the SI although they are not SI units. Table 3 lists the "accepted" units [5]. Some units are fundamental physical quantities, and these are non-SI but can be used with in the SI (Table 4, [6]). There are also a set of non-SI units which are used in certain defined circumstances (Table 5), although they are not necessarily official sanctioned [7].

`\deka` In addition to the units themselves, siunitx provides pre-defined macros for all of the SI

Table 1: SI base units

Unit	Macro	Symbol
ampere	`\ampere`	A
candela	`\candela`	cd
kelvin	`\kelvin`	K
kilogram	`\kilogram`	kg
metre	`\metre`	m
mole	`\mole`	mol
second	`\second`	s

Table 2: Coherent derived units in the SI with special names and symbols

Unit	Macro	Symbol	Unit	Macro	Symbol
becquerel	`\becquerel`	Bq	newton	`\newton`	N
degree Celsius	`\celsius`	°C	ohm	`\ohm`	Ω
coulomb	`\coulomb`	C	pascal	`\pascal`	Pa
farad	`\farad`	F	radian	`\radian`	rad
gray	`\gray`	Gy	siemens	`\siemens`	S
hertz	`\hertz`	Hz	sievert	`\sievert`	Sv
henry	`\henry`	H	steradian	`\steradian`	sr
joule	`\joule`	J	tesla	`\tesla`	T
katal	`\katal`	kat	volt	`\volt`	V
lumen	`\lumen`	lm	watt	`\watt`	W
lux	`\lux`	lx	weber	`\weber`	Wb

Table 3: Non-SI units accepted for use with the International System of Units

Unit	Macro	Symbol
day	`\day`	d
degree	`\degree`	°
hectare	`\hectare`	ha
hour	`\hour`	h
litre	`\litre`	l
	`\liter`	L
minute (plane angle)	`\arcminute`	′
minute (time)	`\minute`	min
percent	`\percent`	%
second (plane angle)	`\arcsecond`	″
tonne	`\tonne`	t

Table 4: Non-SI units whose values in SI units must be obtained experimentally

Unit	Macro	Symbol
astronomical unit	`\astronomicalunit`	ua
atomic mass unit	`\atomicmassunit`	u
bohr	`\bohr`	a_0
speed of light	`\clight`	c_0
dalton	`\dalton`	Da
electron mass	`\electronmass`	m_e
electronvolt	`\electronvolt`	eV
elementary charge	`\elementarycharge`	e
hartree	`\hartree`	E_h
reduced Planck constant	`\planckbar`	$\hbar$

Table 5: Other non-SI units

Unit	Macro	Symbol
ångström	`\angstrom`	Å
bar	`\bar`	bar
barn	`\barn`	b
bel	`\bel`	B
decibel	`\decibel`	dB
knot	`\knot`	kn
millimetre of mercury	`\mmHg`	mmHg
nautical mile	`\nauticalmile`	M
neper	`\neper`	Np

Table 6: SI prefixes

Prefix	Macro	Symbol	Power	Prefix	Macro	Symbol	Power
yocto	\yocto	y	10^{-24}	deca	\deca	da	10^{1}
zepto	\zepto	z	10^{-21}	hecto	\hecto	h	10^{2}
atto	\atto	a	10^{-18}	kilo	\kilo	k	10^{3}
femto	\femto	f	10^{-15}	mega	\mega	M	10^{6}
pico	\pico	p	10^{-12}	giga	\giga	G	10^{9}
nano	\nano	n	10^{-9}	tera	\tera	T	10^{12}
micro	\micro	µ	10^{-6}	peta	\peta	P	10^{15}
milli	\milli	m	10^{-3}	exa	\exa	E	10^{18}
centi	\centi	c	10^{-2}	zetta	\zetta	Z	10^{21}
deci	\deci	d	10^{-1}	yotta	\yotta	Y	10^{24}

prefixes (Table 6, [4]). The spelling "\deka" is provided for US users as an alternative to \deca.

\square
\squared
\cubic
\cubed

A small number of pre-defined powers are provided as macros. \square and \cubic are intended for use before units, with \squared and \cubed going after the unit.

Bq^{2}
$\mathrm{J}^{2}\,\mathrm{lm}^{-1}$
$\mathrm{lx}^{3}\,\mathrm{V}\,\mathrm{T}^{3}$

```
\si{\square\becquerel} \\
\si{\joule\squared\per\lumen} \\
\si{\cubic\lux\volt\tesla\cubed}
```

\tothe
\raiseto

Generic powers can be inserted on a one-off basis using the \tothe and \raiseto macros. These are the only macros for units which take an argument:

H^{5}
$\mathrm{rad}^{4.5}$

```
\si{\henry\tothe{5}} \\
\si{\raiseto{4.5}\radian}
```

\per

Reciprocal powers are indicated using the \per macro. This applies to the next unit only, unless the units/output/sticky per option is turned on.

$\mathrm{J}\,\mathrm{mol}^{-1}\,\mathrm{K}^{-1}$
$\mathrm{J}\,\mathrm{mol}^{-1}\,\mathrm{K}$
H^{-5}
Bq^{-2}

```
\si{\joule\per\mole\per\kelvin} \\
\si{\joule\per\mole\kelvin} \\
\si{\per\henry\tothe{5}} \\
\si{\per\square\becquerel} \\
```

5.1 Creating new macros

The various macro components of a unit have to be defined before they can be used. The package supplies a number of common definitions, but new definitions are also possible.

As the definition of a logical unit should remain the same in a single document, these creation functions are all preamble-only.

`\DeclareSIUnit` New units are produced using the `\DeclareSIUnit` macro. This takes two mandatory arguments, as well as one optional one: `\DeclareSIUnit[`⟨*options*⟩`]{`⟨*unit*⟩`}{`⟨*symbol*⟩`}`. ⟨*symbol*⟩ can contain literal values, other units, multiple prefixes, powers and `\per`, although literal text should not be intermixed with unit macros. The ⟨*options*⟩ argument can be any suitable options, and applies the specific unit macro only. The (first) optional argument to `\SI` and `\si` can be used to override the settings for the unit. A typical example is the `\degree` unit.

3.1415°

```
\SI{3.1415}{\degree} \\
```

This is declared in the package as:

```
\DeclareSIUnit[units/output/number-unit separator={}]
  {\degree}{\SIUnitSymbolDegree}
```

The spacing can still be altered at point of use:

```
\SI{67890}{\degree} \\
\SI[units/output/number-unit separator = \;]{67890}{\degree}
```

67 890°
67 890 °

`\DeclareSIPrefix` `\DeclareSIPrefix*` The standard SI powers of ten are defined by the package, and are described above. However, the user can define new prefixes with `\DeclareSIPrefix`. This has syntax `\DeclareSIPrefix{`⟨*prefix*⟩`}{`⟨*symbol*⟩`}{`⟨*powers-ten*⟩`}`. A starred-version is also available for creating binary prefixes, with the same syntax (⟨*powers-ten*⟩ being replaced by ⟨*powers-two*⟩). For example, `\kilo` and `\kibi` are defined:

```
\DeclareSIPrefix{\kilo}{k}{3}
\DeclareSIPrefix*{\kibi}{Ki}{10}
```

`\DeclareSIPower` `\DeclareSIPower*` Descriptions for powers are created using `\DeclareSIPower`, with syntax `\DeclareSIPower{`⟨*power*⟩`}{`⟨*nu*
This creates a power macro to appear before the unit it applies to. A starred version of the macro creates powers to appear after the unit.

```
%\DeclareSIPower{\quartic}{4}
%\DeclareSIPower*{\totheforth}{4}\\
%\si{\kilogram\totheforth}\\
%\si{\quartic\metre}
```

`\DeclareSIQualifier` Following the syntax of the other macros, qualifiers are created with the syntax `\DeclareSIQualifier{`⟨*qualifier*⟩`}{`⟨*symbol*⟩`}`. In contrast to the other parts of a unit, there are no pre-defined qualifiers. It is therefore entirely up to the user to create these. For example, to identify the mass of a product created when using a particular catalyst, the preamble could contain:

```
\DeclareSIQualifier{\product}{prod}
\DeclareSIQualifier*{\catalyst}{cat}
```

and then in the body the document could read:

```
\SI{1.234}{\gram\product\per\mole\catalyst\per\hour}
```

$1.234\,\mathrm{g_{prod}\,mol_{cat}^{-1}\,h^{-1}}$

6 The key–value control system

`\sisetup` The behaviour of the siunitx package is controlled by a number of key–value options. These can be given globally using the `\sisetup` function or locally as the optional argument to the user macros.

All of the keys are controlled using the pgfkeys approach to organisation. This means that the keys are split into "paths" of related keys. A single key is set by giving the path plus key name; if you need to set several keys on the same path, you can "change" to the appropriate path and give the key name alone. For example, valid numerical input is controlled by keys in the `numbers/input/` path:

```
\sisetup{
  numbers/input/signs            = +-\pm\mp,
  numbers/input/exponent markers = dDeE
}
```

or

```
\sisetup{
  numbers/input/.cd,
  signs            = +-\pm\mp,
  exponent markers = dDeE
}
```

are both valid.

The package uses a range of different key types:

`Choice` Takes a limited number of choices, which are described separately for each key.

`Literal` A key which uses the value(s) given directly, either to check input (for example the `numbers/input` keys) or in output.

`Maths` Similar to a `literal` option, but the input is always used in maths mode, irrespective of other siunitx settings. Thus to text-mode only input must be placed inside the argument of a `\text` macro.

`Macro` Requires a macro, which may need a single argument.

Table 7: font/detect/ options

Option name	Type	Default
all	Style	⟨*none*⟩
bold	Switch	false
display maths	Switch	false
family	Switch	false
inline bold	Choice	text
italic	Switch	false
mode	Switch	false
none	Style	⟨*none*⟩

Style A key which contains a number of other keys to set. Only the key name should be given: no value is required. This type of key is user-definable, as described in Section **??**.

Switch These are on–off switches, and recognise true, on and yes to turn on, and false, off and no to turn off. Giving just the key name also turns the key on.

The tables of option names use these descriptions to indicate how the keys should be used.

In all cases, UK and US English spellings are available for both option names and for settings. Thus centre and center can be used for alignment options, and maths or math is valid in the names of font options. In the rest of this document, UK English spelling is used.

6.1 Detecting fonts

The siunitx package controls the font used to print output independently of the surrounding material. The standard method is to ignore the surroundings entirely, and to use the current upright maths font for all printing.[4] However, the package can detect and follow surrounding bold, italic and font family changes. The font detection options are available in path font/detect/ and are summarised in Table 7.

font/detect/bold
font/detect/family
font/detect/italic
font/detect/mode

The four basic options bold and italic set detection of the prevailing bold and italic states, respectively. The italic state is only checked if the surrounding material is not in maths mode (as maths text is always italic). Detecting the current family (roman, sans serif or monospaced) is controlled by the family setting, while the current mode (text or maths) is detected using the mode switch.

font/detect/all
font/detect/none

The two style options all and none can be used to turn on or off all of the detection functions in one go. These are style options, and so need no value.

[4] This will typically use \mathrm.

1234
1234
1234
1234
1234
1234
1234
1234
1234
1234

```
\sisetup{font/detect/none}%
$\num{1234}$ \\
\num{1234} \\
\emph{\num{1234}} \\
\textbf{\num{1234}} \\
\textbf{$\num{1234}$} \\
\sisetup{font/detect/all}%
$\num{1234}$ \\
\num{1234} \\
\emph{\num{1234}} \\
\textbf{\num{1234}} \\
\textbf{$\num{1234}$} \\\
```

`font/detect/inline bold` Bold detection is influenced by the value of `inline bold`, which takes values `text` and `maths`. The package can detect the local value of bold for either the surrounding text, or the surrounding inline (`$...$`) maths.

1234
1234
1234
1234
1234

```
\sisetup{
  font/detect/bold = on,
  font/detect/inline bold = maths
}%
$\num{1234}$ \\
{ \boldmath $\num{1234}$ } \\
{ \bfseries $\num{1234}$ } \\
\sisetup{
  font/detect/inline bold = text
}
{ \boldmath $\num{1234}$ } \\
{ \bfseries $\num{1234}$ }
```

`font/detect/display maths` The font detection system can treat displayed mathematical content in two ways. This is controlled by the `display` option. When set `on`, display mathematics is treated independently from the body of the document. Thus the local *maths* font is checked for matching. In contrast, when set `off`, display material is treated with the current running text font.

```
\sffamily
Some text
\sisetup{
  font/detect/all,
  font/detect/display maths = true
}
\[ x = \SI{1.2e3}{\kilogram\kelvin\candela} \]
More text
\sisetup{font/detect/display maths = false}
\[ y = \SI{3}{\metre\second\mole} \]
```

Table 8: `font/` options (all also apply in `font/units/` and `font/numbers`)

Option name	Type	Default
`maths rm`	Macro	`\mathrm`
`maths sf`	Macro	`\mathsf`
`maths tt`	Macro	`\mathtt`
`mode`	Choice	`maths`
`text rm`	Macro	`\rmfamily`
`text sf`	Macro	`\sffamily`
`text tt`	Macro	`\ttfamily`

Some text

$$x = 1.2 \times 10^{3}\,\mathrm{kg\,K\,cd}$$

More text

$$y = 3\,\mathrm{m\,s\,mol}$$

6.2 Output font families

The relationship between font family detected and font family used for output is not fixed. The font detected by the package in the surrounding material does not have to match that used for output. This is controlled by the `font/output` options.

`font/mode` The `mode` option determines whether siunitx uses maths or text mode when printing output. The choices are `maths`, `math` and `text`. When using maths mode, text is printed using a maths font whereas in text mode a text font is used. The extent to which this is visually obvious depends on the fonts in use in the document. This manual uses old style (lower-case) figures in text mode to highlight the differences. This option has no effect if the `font/detect/mode` switch is `on`.

`font/maths rm`
`font/text rm`
`font/maths sf`
`font/maths tt`
`font/text sf`
`font/text tt`

If font family detection is inactive, siunitx uses the font family stored in either `maths rm` or `text rm` for output. The choice of `maths` or `text` depends on the `mode` setting. If font family detection is active, siunitx may be using a sans serif or monospaced font for output. In maths mode, these are stored in `maths sf` and `maths tt`, and for text mode in `text sf` and `text tt`. Notice that the detected and output font families can differ.

```
\sisetup{font/detect/family}%
\num{1234} \\
{ \sffamily \num{1234} } \\
\SI{99}{\metre} \\
\sisetup{font/maths rm = \mathtt}%
\SI{99}{\metre}
```

1234
1234
99 m
99 m

This can be used to good effect to change all output from siunitx without needing to detect the font. For example, when creating beamer presentations the settings

Table 9: numbers/input/ options

Option name	Type	Default
complex roots	Literal	ij
close uncertainty	Literal	)
decimal markers	Literal	.,
digits	Literal	0123456789
exponent markers	Literal	dDeE
ignore	Literal	⟨*none*⟩
open uncertainty	Literal	(
signs	Literal	+-\pm\mp
symbols	Literal	\pi

```
\sisetup{
  font/maths rm = \mathsf,
  font/text rm  = \sffamily
}
```

given all output in sans serif font without font detection.

Every one of the font options can be given independently for units and number, with the option paths font/units/ and font/numbers/, respectively. This allows fine control of output.

6.3 Parsing numbers

The package uses a sophisticated parsing system to understand numbers. This allows siunitx to carry out a range of formatting, as described later. All of the input options take lists of literal tokens, and are summarised in Table 9.

numbers/input/digits
numbers/input/decimal markers
numbers/input/signs
numbers/input/exponent markers

The basic parts of a number are the digits, any sign and a separator between the integer and decimal parts. These are stored in the input options digits, decimal markers and signs, respectively. More than one input decimal marker can be used: it will be converted by the package to the appropriate output marker. Numbers which include an exponent part also require a marker for the exponent: this again is taken from the range of tokens in the exponent markers option.

numbers/input/ignore
numbers/input/symbols

As well as "normal" digits, the package will interpret symbolic "numbers" (such as \pi) correctly if they are included in the symbols list. Tokens given in the ignore list are totally passed over by siunitx: they will be removed from the input with no further processing.

numbers/input/open uncertainty
numbers/input/close uncertainty

In some fields, it is common to give the uncertainty in a value in brackets after the main part of the number, for example "1.234(5)". The opening and closing symbols used for this type of input are set as open uncertainty and close uncertainty.

numbers/input/complex roots

When using complex numbers in input, the complex root ($\sqrt{-1}$) is indicated by one of

Table 10: numbers/process/ options

Option name	Type	Default
add zero decimal	Switch	false
add zero integer	Switch	false
explicit sign	Literal	+
include explicit sign	Switch	false
retain explicit plus	Switch	false
retain zero exponent	Switch	false
round mode	Choice	off
round figures	Number	2
round places	Number	2

the tokens stored in complex roots.

6.4 Post-processing numbers

Before typesetting numbers, various post-processing steps can be carried out. These involve adding or removing information from the number in a systematic way; the options are summarised in Table 10.

numbers/process/round mode
numbers/process/round figures
numbers/process/round places

The siunitx package can round numerical input to a fixed number of significant figures or decimal places. This is controlled by the round mode option, which takes the choices off, figures and places. When rounding is turned on, the number of figures to use is determined by the round figures and round places option: both of these options require a number.

```
\num{1.23456} \\
\num{14.23} \\
\sisetup{
  numbers/process/round mode   = places,
  numbers/process/round places = 3
}%
\num{1.23456} \\
\num{14.23} \\
\sisetup{
  numbers/process/round mode    = figures,
  numbers/process/round figures = 3
}%
\num{1.23456} \\
\num{14.23}
```

1.234 56
14.23
1.235
14.230
1.23
14.2

numbers/process/add zero decimal
numbers/process/add zero integer

It is possible to give real (floating point) numbers as input omitting the decimal or the integer parts of the number (for example 0.123 or 123.0). The options `add zero decimal` and `add zero integer` allow the package to "fill in" the missing zero.

```
\num{123.} \\
\num{.456} \\
\sisetup{
  numbers/process/add zero decimal = no,
  numbers/process/add zero integer = no,
}%
\num{123.} \\
\num{.456}
```

123.0
0.456
123.
.456

numbers/process/explicit sign
numbers/process/include explicit sign
numbers/process/retain explicit plus

The inclusion of a leading plus sign is usually unnecessary for positive numbers, and so the `retain explicit plus` option is available to control whether these are printed. As the same time, it may be useful to force all numbers to have a sign. This behaviour is controlled by the `include explicit sign` option, with the sign to use stored by the `explicit sign` option.

```
\num{+345} \\
\num[numbers/process/retain explicit plus]{+345} \\
\num[
  numbers/process/explicit sign = -,
  numbers/process/include explicit sign,
]{345}
```

345
+345
−345

numbers/process/retain zero exponent

The retention of a zero exponent (10^0) is controlled by the `retain zero exponent` option.

```
\num{444e0} \\
\num[numbers/process/retain zero exponent = yes]{444e0}
```

444
444×10^{0}

Table 11: numbers/output/ options

Option name	Type	Default
close bracket	Literal	)
close uncertainty	Literal	)
complex root	Maths	i
decimal marker	Maths	.
exponent base	Literal	10
exponent product	Maths	\times
group digits	Switch	true
group four digits	Switch	false
group separator	Maths	\,
open bracket	Literal	(
open uncertainty	Literal	(
separate uncertainty	Switch	false
tight spacing	Switch	false
use brackets	Switch	true
uncertainty space	Maths	⟨*none*⟩

6.5 Printing numbers

Actually printing numbers is controlled by a number of settings, which apply ideas such as differing decimal markers, digit grouping and so on. All of these options are concerned with the appearance of output, rather than the data it conveys. The options are summarised in Table 13.

numbers/output/group digits
numbers/output/group four digits
numbers/output/group separator

Grouping digits into blocks of three is a common method to increase the ease of reading of numbers. The group digits choice turns this behaviour on and off, with grouping for numbers of exactly four digits controlled by the group four digits choice. Note that the later only applies if group digits is turned on. The separator used between groups of digits is stored by the group separator option. This takes literal input and is used in maths mode: for a text-mode full space use \text{~}.

```
\num{12345} \\
\num[numbers/output/group digits = off]{12345} \\
\num{1234} \\
\num[numbers/output/group four digits = on]{1234} \\
\num{12345} \\
\num[numbers/output/group separator = {{{,}}}]{12345} \\
\num[numbers/output/group separator = \text{~}]{12345}
```

12 345
12345
1234
1 234
12 345
12,345
12 345

numbers/output/complex root
numbers/output/decimal marker

The decimal marker used in output is set using the `decimal marker` option. This can differ from the input marker, as can the root of $\sqrt{-1}$, which is stored in the `complex root` option. The later is always in maths mode, but notice that siunitx uses `\mathrm` by default. To force standard TEX maths output to give i, the construction `\text{\ensuremath{i}}` can be used.

```
\num{1.23} \\
\num[numbers/output/decimal marker = {{{,}}}]{1.23} \\
\num[numbers/output/complex root = \text{\ensuremath{i}}]{1+2i}
```

1.23
1,23
$1 + 2i$

numbers/output/exponent base
numbers/output/exponent product

When exponents are present in the input, the options `exponent base` and `exponent product` set the obvious parts of the output. Notice that the base is in the current mode, but the product sign is always in maths mode.

```
\num[numbers/output/exponent product = \times]{1e2} \\
\num[numbers/output/exponent product = \cdot]{1e2} \\
\num[numbers/output/exponent base = 2]{1e2}
```

1×10^{2}
$1 \cdot 10^{2}$
1×2^{2}

numbers/output/separate uncertainty
numbers/output/uncertainty space
numbers/output/open uncertainty
numbers/output/close uncertainty

When input is given including an uncertatinty in a value, it can be printed either with the uncertainty in brackets or as a separate number. This behaviour is controlled by the `separate uncertainty` choice. If the uncertainty is given in brackets, a space may be added between the main value and the uncertainty: this is stored using the `uncertainty space` option. The opening and closing brackets used are stored `open uncertainty` and `close uncertainty`, respectively.

```
\num{1.234(5)} \\
\num[numbers/output/separate uncertainty = on]{1.234(5)} \\
\sisetup{
  numbers/output,
  open uncertainty  = [,
  close uncertainty = ],
  uncertainty space = {\,}
}
\num{1.234(5)}
```

1.234(5)
1.234 ± 0.005
1.234 [5]

numbers/output/use brackets
numbers/output/open bracket
numbers/output/close bracket

There are certain combinations of numerical input which can be ambiguous. This can be corrected by adding brackets in the appropriate place, and is controlled by the `use brackets` switch. The opening and closing brackets used are stored `open bracket` and `close bracket`, respectively.

```
\num{1+2i e10} \\
\num[numbers/output/use brackets = false]{1+2i e10} \\
\sisetup{
  numbers/output,
  open bracket  = \{,
  close bracket = \},
}
\num{1+2i e10}
```

$(1 + 2\mathrm{i}) \times 10^{10}$
$1 + 2\mathrm{i} \times 10^{10}$
$\{1 + 2\mathrm{i}\} \times 10^{10}$

numbers/output/tight spacing

Under some circumstances is may be desirable to "squeeze" the output spacing. This is turned on using the `tight spacing` switch, which compresses spacing where possible.

```
\num{1\pm2i e3} \\
\num[numbers/output/tight spacing = true]{1\pm2i e3} \\
```

$(1 \pm 2\mathrm{i}) \times 10^{3}$
$(1{\pm}2\mathrm{i}){\times}10^{3}$

6.6 Creating units

The various macro units are created at the start of the document. These can be defined only inside the `\si` and `\SI` macros, or can also be made available in the document body. There are a number of settings which control this creation process (Table 12). As a result, these options all apply in the preamble only.

units/creation/allow in body
units/creation/overwrite macros

The `allow in body` option controls whether the unit macros exist outside of the `\si` and `\SI` arguments. When this option is `true`, siunitx creates the macros for general use. The standard method to achieve this does not overwrite any existing macros: this behaviour can be altered using the `overwrite macros` switch.

units/creation/add leading space
units/creation/allow optional argument
units/creation/use xspace

When "free standing" unit macros are created, their behaviour can be adjusted by a number of options. These are mainly intended for emulating the input syntax of older packages. The option `allow optional argument` gives the same behaviour for the inputs

Table 12: units/creation/ options

Option name	Type	Default
add leading space	Switch	false
allow in body	Switch	false
allow optional argument	Switch	false
overwrite macros	Switch	false
use xspace	Switch	false

```
\SI{10}{\metre}
```

and

```
\metre[10].
```

The add leading space and use xspace options control the behaviour at the "ends" of the unit macros. Activating add leading space inserts the number–unit space before the unit is printed. This is suitable for the input syntax

```
30\metre
```

but does mean that the unit macros are incorrectly spaced in running text. On the other hand, the use xspace option attempts to correctly space input such as

```
\metre is the symbol for metres.
```

6.7 Using units

Part of the power of siunitx is the ability to alter the output format for units without changing the input. The behaviour of units is therefore controlled by a number of options which alter either the processing of units or the output directly.

units/output/allow literal units

Some users may prefer to completely disable the use of literal input in units, for example to enforce consistency. This can be accomplished by setting the allow literal units switch. With this option enabled, only macro-based units can be used in a document.

units/output/allow unit-number breaks

The standard method for printing units with a value prevents a break occurring between the two parts. The allow unit-number breaks can be used to turn on breaking in this context, for example when using narrow columns.

```
\begin{minipage}{4cm}
  Some text to act as filler \SI{10}{\kilogram\cubic\metre} \\
  \sisetup{units/output/allow number-unit breaks}%
  Some text to act as filler \SI{10}{\kilogram\cubic\metre}
\end{minipage}
```

Table 13: `units/output` options

Option name	Type	Default
`allow literal units`	Switch	`true`
`allow number-unit breaks`	Switch	`false`
`close bracket`	Literal	`)`
`inter-unit separator`	Literal	`\,`
`inter-unit space`	Literal	`\,`
`number-unit separator`	Literal	`\,`
`open bracket`	Literal	`(`
`print as fraction`	Switch	`false`
`qualifier format`	Choice	`subscript`
`sticky per`	Switch	`false`
`use brackets`	Switch	`true`

Some text to act as filler $10\,\mathrm{kg\,m^3}$
Some text to act as filler 10 $\mathrm{kg\,m^3}$

`units/output/inter-unit separator`
`units/output/number-unit separator`
`units/output/inter-unit space`

The separation between units, and between a number and the units following it, can be adjusted. The options `inter-unit separator` and `number-unit separator` are used to store an appropriate spacing or separator value. The separators are always printed in maths mode: a full text space is therefore given as `\text{ }`. When using literal unit input, the `~` symbol is converted into an inter-unit space which can be different from that specified in `inter-unit separator`.

`units/output/sticky per`

By default, `\per` applies only to the next unit given.[5] By setting the `sticky per` flag, this behaviour is changed so that `\per` applies to all subsequent units.

$\mathrm{Pa\,Gy^{-1}\,H}$
$\mathrm{Pa\,Gy^{-1}\,H^{-1}}$

```
\si{\pascal\per\gray\henry} \\
\si[units/output/sticky per]
  {\pascal\per\gray\henry}
```

`units/output/use brackets`
`units/output/open bracket`
`units/output/close bracket`

As with numerical output, there are some occasions when units can potentially be ambiguous. This can be corrected by adding brackets in the appropriate place, and is controlled by the `use brackets` switch. The opening and closing brackets used are stored `open bracket` and `close bracket`, respectively.

[5] This is the standard method of reading units in English: for example, $\mathrm{J\,mol^{-1}\,K^{-1}}$ is pronounced "joules per mole per kelvin".

```
\sisetup{units/output/qualifier format = space}%
\SI{1.234}{\gram\product\cubed\per\mole} \\
\SI[units/output/use brackets = false]
  {1.234}{\gram\product\cubed\per\mole} \\
\SI[units/output/open bracket = \{,
    units/output/close bracket = \}]
  {1.234}{\gram\product\cubed\per\mole}
```

$1.234\,(\mathrm{g\ prod})^{3}\,\mathrm{mol}^{-1}$
$1.234\,\mathrm{g\ prod}^{3}\,\mathrm{mol}^{-1}$
$1.234\,\{\mathrm{g\ prod}\}^{3}\,\mathrm{mol}^{-1}$

`units/output/qualifier format` Unit qualifiers can appear in a number of output formats. The package expects a choice from the list `subscript`, `space` and `brackets`. These work as might be expected.

```
\si[units/output/qualifier format = subscript]{\gram\product} \\
\si[units/output/qualifier format = space]{\gram\product} \\
\si[units/output/qualifier format = brackets]{\gram\product}
```

$\mathrm{g}_{\mathrm{prod}}$
$\mathrm{g\ prod}$
$\mathrm{g(prod)}$

When the qualifier is given as a subscript, no brackets will be added if a power is given for the same unit. For other formats, brackets are added if `use brackets` is true.

```
\si[units/output/qualifier format = subscript]
  {\gram\product\squared} \\
\si[units/output/qualifier format = space]
  {\gram\product\squared}
```

$\mathrm{g}^{2}_{\mathrm{prod}}$
$(\mathrm{g\ prod})^{2}$

6.8 Symbols

Most units use letters as the symbol for the unit, and these are all very easy to control. However, a small number of units use other symbols, and matching these to the body text requires more work. siunitx provides appropriate symbols for commonly-used units, but the definitions may need adjustment depending on the body font used in a document.

`symbols/redefine symbols` The package provides one general option for the handling of symbols. If the packages textcomp or upgreek are loaded, symbols can be taken from these for units, rather than using the siunitx default values. The switch `redefine symbols` can be used to turn this behaviour on or off: the standard setting is `true`.

The individual symbols used for a number of units are available as package options in paths `symbols/maths` and `symbols/text`. The option names are summarised in Table 14.

Table 14: symbols/maths and symbols/text options

Option name	Type
angstrom	Literal
arcminute	Literal
arcsecond	Literal
celsius	Literal
degree	Literal
micro	Literal
ohm	Literal

The standard values can be somewhat involved, and the interested reader is directed to the code for full details. In many cases, the maths and text variations are created from the same underlying macro using \text or \ensuremath. For example, the standard definitions for angstrom are:

```
\sisetup{
  symbols/maths/angstrom = \text{\AA},
  symbols/text/angstrom = \AA,
}
```

\SIUnitSymbolAngstrom
\SIUnitSymbolArcminute
\SIUnitSymbolArcsecond
\SIUnitSymbolCelsius
\SIUnitSymbolDegree
\SIUnitSymbolMicro
\SIUnitSymbolOhm

The maths and text symbols defined above are wrapped up into mode independent functions with user names. These are then used in the definitions of the appropriate units. For example, the angstrom symbol can be accessed using the macro \SIUnitSymbolAngstrom. Notice that these names capitalise the unit name (to make reading the macro name easier!).

7 Usage tips and known issues

7.1 Ensuring text or maths output

The macros \ensuremath and \text should be used to ensure that a particular item is always printed in the desired mode. Some mathematical output does not work well in \mathrm (the standard font used by siunitx for printing). The easiest way to solve this is to use the construction \text{\ensuremath{...}}, which will print the material in the standard mathematics font without affecting the rest of the output.

```
\num[numbers/output/complex root = i]{1+2i} \\
\num[numbers/output/complex root = \ensuremath{i}]{1+2i} \\
\num[numbers/output/complex root = \text{\ensuremath{i}}]{1+2i}
```

$1+2\mathrm{i}$
$1+2\mathrm{i}$
$1+2i$

In some circumstances, forcing `\mathnormal` may suffice, but this deals less well with non-Latin characters.

7.2 Using a comma as a separator

The use of a comma as a group separator or decimal marker is common in many parts of the world. This is easily achieved in siunitx, but does require three sets of braces in the option to give the normal spacing.

```
\num[numbers/output/decimal marker = {,}]{1.23} \\     % Wrong
\num[numbers/output/decimal marker = {{,}}]{1.23} \\   % Wrong
\num[numbers/output/decimal marker = {{{,}}}]{1.23} \\ % Correct
```

1,23
1,23
1,23

Change History

v1.0
General: First official release 1
v1.1
General: Package extended to a greater range of unit types 1
v1.2
General: Correct handling for ranges of numbers added 1
v2.0
General: Complete re-write of package to add many new features 1

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A

\ampere 5
\angstrom 7
\arcminute 7
\arcsecond 7
\astronomicalunit 7
\atomicmassunit 7
\atto 8

B

\bar 7
\barn 7
\becquerel 6
\bel 7

\bohr ... 7

C

\candela ... 5
\celsius ... 6
\centi ... 8
\clight ... 7
\coulomb ... 6
\cubed ... 6
\cubic ... 6

D

\dalton ... 7
\day ... 7
\deca ... 8
\deci ... 8
\decibel ... 7
\DeclareSIPower ... 9
\DeclareSIPower* ... 9
\DeclareSIPrefix ... 8
\DeclareSIPrefix* ... 8
\DeclareSIQualifier ... 9
\DeclareSIUnit ... 8
\degree ... 7
\deka ... 6

E

\electronmass ... 7
\electronvolt ... 7
\elementarycharge ... 7
\exa ... 8

F

\farad ... 6
\femto ... 8
font/detect/all(option) ... 11
font/detect/bold(option) ... 10
font/detect/display maths(option) ... 12
font/detect/family(option) ... 10
font/detect/inline bold(option) ... 11
font/detect/italic(option) ... 10
font/detect/mode(option) ... 10
font/detect/none(option) ... 11
font/maths rm(option) ... 13
font/maths sf(option) ... 13
font/maths tt(option) ... 13
font/mode(option) ... 12
font/text rm(option) ... 13
font/text sf(option) ... 13
font/text tt(option) ... 13

G

\giga ... 8
\gray ... 6

H

\hartree ... 7
\hectare ... 7
\hecto ... 8
\henry ... 6
\hertz ... 6
\hour ... 7

J

\joule ... 6

K

\katal ... 6
\kelvin ... 5
\kilo ... 8
\kilogram ... 5
\knot ... 7

L

\liter ... 7
\litre ... 7
\lumen ... 6
\lux ... 6

M

\mega ... 8
\meter ... 5
\metre ... 5
\micro ... 8
\milli ... 8
\minute ... 7
\mmHg ... 7
\mole ... 5

N

\nano ... 8
\nauticalmile ... 7
\neper ... 7
\newton ... 6
\num ... 3
numbers/input/close uncertainty(option) ... 14
numbers/input/complex roots(option) ... 14
numbers/input/decimal markers(option) ... 13
numbers/input/digits(option) ... 13
numbers/input/exponent markers(option) ... 13
numbers/input/ignore(option) ... 14

numbers/input/open uncertainty(option) . . . 14
numbers/input/signs(option) . . . 13
numbers/input/symbols(option) . . . 14
numbers/output/close bracket(option) . 18
numbers/output/close uncertainty(option) . . . 18
numbers/output/complex root(option) . 17
numbers/output/decimal marker(option) 17
numbers/output/exponent base(option) . 17
numbers/output/exponent product(option) . . . 17
numbers/output/group digits(option) . 16
numbers/output/group four digits(option) . . . 16
numbers/output/group separator(option) . . . 16
numbers/output/open bracket(option) . 18
numbers/output/open uncertainty(option) . . . 18
numbers/output/separate uncertainty(option) . . . 18
numbers/output/tight spacing(option) . 18
numbers/output/uncertainty space(option) . . . 18
numbers/output/use brackets(option) . 18
numbers/process/add zero decimal(option) . . . 15
numbers/process/add zero integer(option) . . . 15
numbers/process/explicit sign(option) 16
numbers/process/include explicit sign(option) . . . 16
numbers/process/retain explicit plus(option) . . . 16
numbers/process/retain zero exponent(option) . . . 16
numbers/process/round figures(option) 14
numbers/process/round mode(option) . . 14
numbers/process/round places(option) . 14

O

\ohm . . . 6
options:
font/detect/all . . . 11
font/detect/bold . . . 10
font/detect/display maths . . . 12
font/detect/family . . . 10
font/detect/inline bold . . . 11
font/detect/italic . . . 10
font/detect/mode . . . 10
font/detect/none . . . 11
font/maths rm . . . 13
font/maths sf . . . 13
font/maths tt . . . 13
font/mode . . . 12
font/text rm . . . 13
font/text sf . . . 13
font/text tt . . . 13
numbers/input/close uncertainty . 14
numbers/input/complex roots . . . 14
numbers/input/decimal markers . . . 13
numbers/input/digits . . . 13
numbers/input/exponent markers . . 13
numbers/input/ignore . . . 14
numbers/input/open uncertainty . . 14
numbers/input/signs . . . 13
numbers/input/symbols . . . 14
numbers/output/close bracket 18
numbers/output/close uncertainty 18
numbers/output/complex root 17
numbers/output/decimal marker . . . 17
numbers/output/exponent base 17
numbers/output/exponent product . 17
numbers/output/group digits 16
numbers/output/group four digits 16
numbers/output/group separator . . 16
numbers/output/open bracket 18
numbers/output/open uncertainty . 18
numbers/output/separate uncertainty . . . 18
numbers/output/tight spacing 18
numbers/output/uncertainty space 18
numbers/output/use brackets 18
numbers/process/add zero decimal 15
numbers/process/add zero integer 15
numbers/process/explicit sign . . . 16
numbers/process/include explicit sign . . . 16
numbers/process/retain explicit plus . . . 16
numbers/process/retain zero exponent . . . 16
numbers/process/round figures . . . 14
numbers/process/round mode 14
numbers/process/round places 14
symbols/maths/angstrom . . . 22
symbols/maths/arcminute . . . 22
symbols/maths/arcsecond . . . 22
symbols/maths/celsius . . . 22

symbols/maths/degree 22
symbols/maths/micro 22
symbols/maths/ohm 22
symbols/redefine symbols 22
symbols/text/angstrom 22
symbols/text/arcminute 22
symbols/text/arcsecond 22
symbols/text/celsius 22
symbols/text/degree 22
symbols/text/micro 22
symbols/text/ohm 22
units/creation/add leading space 19
units/creation/allow in body 19
units/creation/allow optional argument 19
units/creation/overwrite macros . 19
units/creation/use xspace 19
units/output/allow literal units 20
units/output/allow unit-number breaks 20
units/output/close bracket 21
units/output/inter-unit separator 20
units/output/inter-unit space . . . 20
units/output/number-unit separator 20
units/output/open bracket 21
units/output/qualifier format . . . 21
units/output/sticky per 20
units/output/use brackets 21

P

\pascal 6
\per 6
\percent 7
\peta 8
\pico 8
\planckbar 7

R

\radian 6
\raiseto 6

S

\second 5
\SI 4
\si 4
\siemens 6
\sievert 6
\sisetup 9
\SIUnitSymbolAngstrom 22
\SIUnitSymbolArcminute 22
\SIUnitSymbolArcsecond 22
\SIUnitSymbolCelsius 22
\SIUnitSymbolDegree 22
\SIUnitSymbolMicro 22
\SIUnitSymbolOhm 22
\square 6
\squared 6
\steradian 6
symbols/maths/angstrom(option) 22
symbols/maths/arcminute(option) 22
symbols/maths/arcsecond(option) 22
symbols/maths/celsius(option) 22
symbols/maths/degree(option) 22
symbols/maths/micro(option) 22
symbols/maths/ohm(option) 22
symbols/redefine symbols(option) 22
symbols/text/angstrom(option) 22
symbols/text/arcminute(option) 22
symbols/text/arcsecond(option) 22
symbols/text/celsius(option) 22
symbols/text/degree(option) 22
symbols/text/micro(option) 22
symbols/text/ohm(option) 22

T

\tera 8
\tesla 6
\tonne 7
\tothe 6

U

units/creation/add leading space(option) 19
units/creation/allow in body(option) . 19
units/creation/allow optional argument(option) 19
units/creation/overwrite macros(option) 19
units/creation/use xspace(option) . . . 19
units/output/allow literal units(option) 20
units/output/allow unit-number breaks(option) 20
units/output/close bracket(option) . . 21
units/output/inter-unit separator(option) 20
units/output/inter-unit space(option) 20

units/output/number-unit separator(option) 20
units/output/open bracket(option) . . . 21
units/output/qualifier format(option) 21
units/output/sticky per(option) 20
units/output/use brackets(option) . . . 21

V
\volt 6

W
\watt 6
\weber 6

Y
\yocto 8
\yotta 8

Z
\zepto 8
\zetta 8

References

[1] *The International System of Units (SI)*, http://www.bipm.org/en/si/.

[2] *SI base units*, http://www.bipm.org/en/si/si_brochure/chapter2/2-1/.

[3] *Units with special names and symbols; units that incorporate special names and symbols*, http://www.bipm.org/en/si/si_brochure/chapter2/2-2/2-2-2.html.

[4] *SI Prefixes*, http://www.bipm.org/en/si/si_brochure/chapter3/prefixes.html.

[5] *Non-SI units accepted for use with the International System of Units*, http://www.bipm.org/en/si/si_brochure/chapter4/table6.html.

[6] *Non-SI units whose values in SI units must be obtained experimentally*, http://www.bipm.org/en/si/si_brochure/chapter4/table7.html.

[7] *Other non-SI units*, http://www.bipm.org/en/si/si_brochure/chapter4/table8.html.